CS 4410
Operating Systems

Networking:
Transport Layer

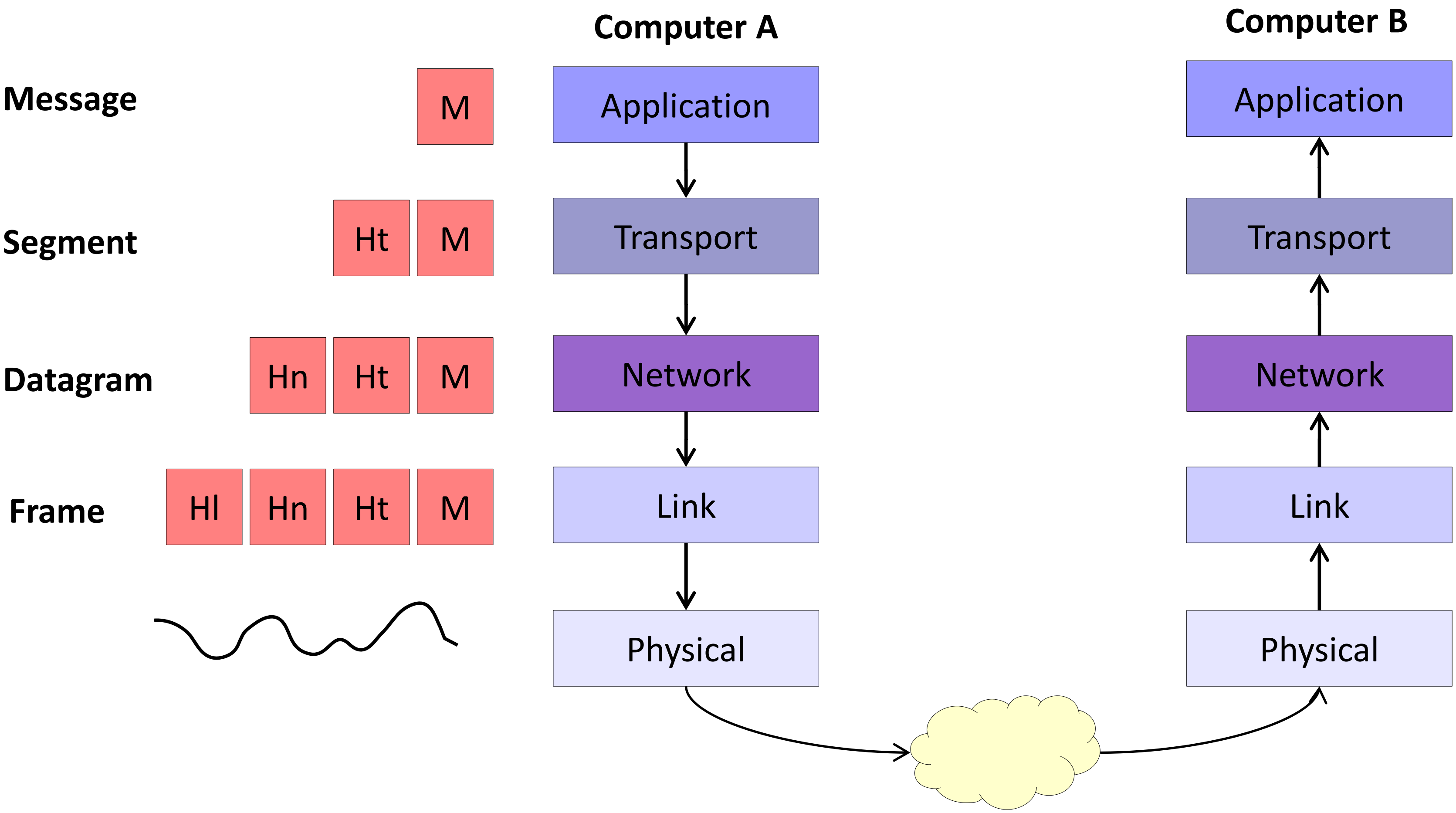
Summer 2016

Cornell University

Today

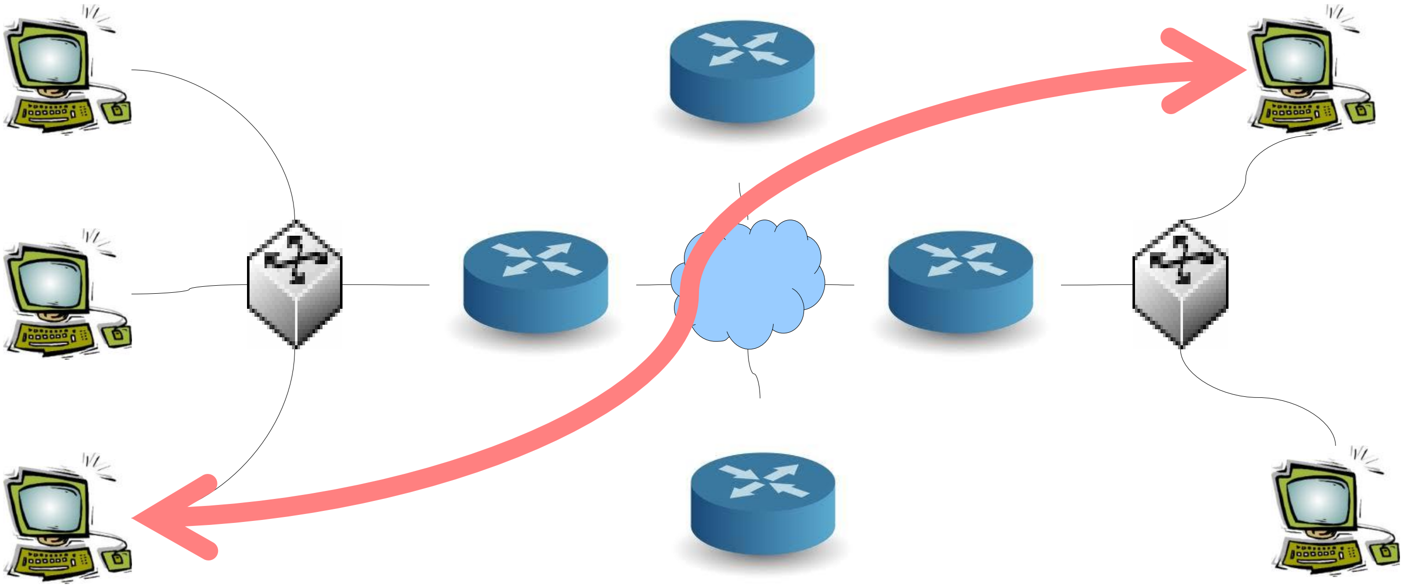
- Logical communication between remote processes.

Protocol Stack



Transport Layer

- It offers **logical communication** between processes.
- Networking processes think that they directly speak to each other.



Transport Layer

- Mission: Transfer a segment from one process to another.
- Services:
    - Multiplexing – Demultiplexing
    - Error Detection
    - Reliable data transfer
        - It takes care of packet loss and reordering.
- Examples: UDP, TCP
    - UDP offers the first two services and TCP offers all the services.
- Transport Layer Protocols are implemented only at terminal nodes (computers).
    - Routers and switches do not implement this layer.

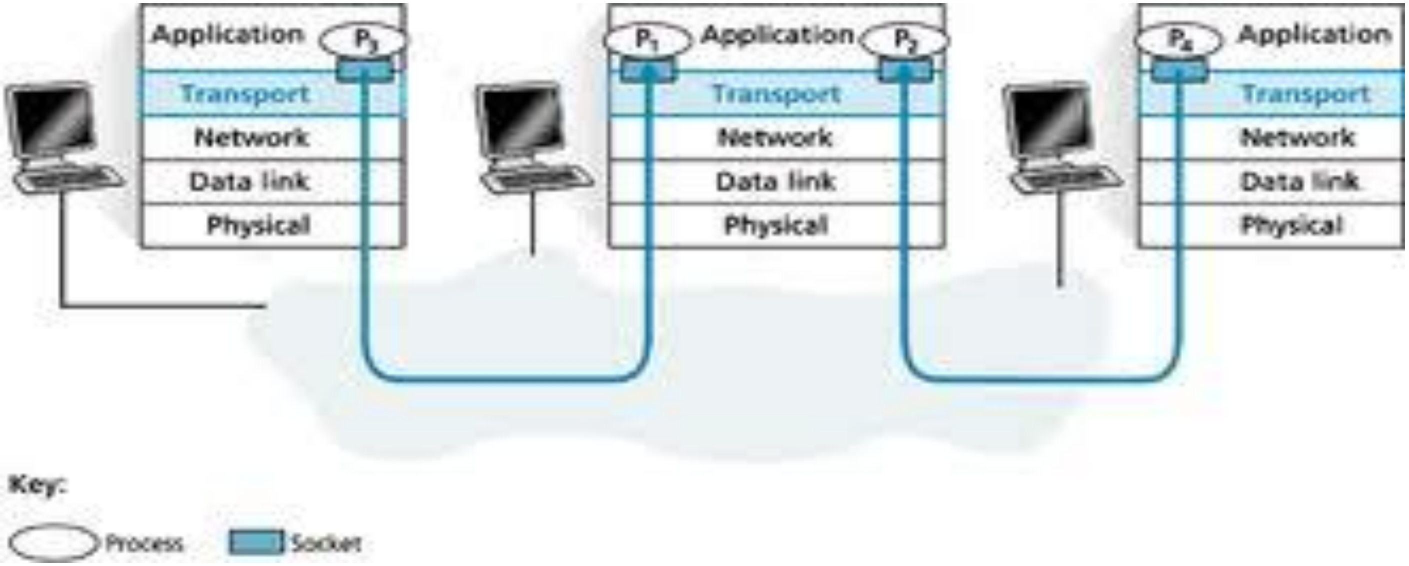
Multiplexing - Demultiplexing

- How does the Transport Layer know to which process it should forward the received data?
- How does the Transport Layer collect the data that processes want to send and forward it to the network layer?
- Each process can create one or more **sockets**.
  - Processes see sockets as the only **gateway to the network**. They can send or receive data only through them.
  - In reality, they are **structures of the OS** that maintain valuable information for the connection.
  - One field of the socket is its port number, a unique id in the system.
  - Sockets are like file descriptors. When a process wants to send data, it invokes a system call, passing the socket and the pointer to data.

Multiplexing - Demultiplexing

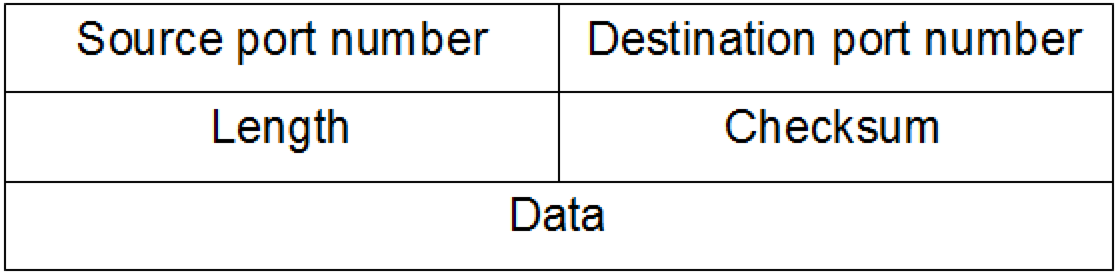
- From the “other side” of the socket there is the **Transport layer,** implemented in the OS.
- The Transport Layer:
  - takes the data from the process,
  - splits the data into frames,
  - reads the fields of the corresponding socket,
  - creates the header for each frame (being based on the fields)
  - and forwards the frame to the Network layer.
- This process is called **Multiplexing**.
- When a frame is forwarded from the Network Layer to the Transport Layer, the latter checks the header, identifies the port number of the socket-destination and forwards the data there.
- This process is called **Demultiplexing**.

Multiplexing - Demultiplexing



UDP

- User Datagram Protocol
- Services: Multiplexing-Demultiplexing, Error detection.
- It is so light that the process “roughly” talks directly to the Network layer.
- It is not reliable, but it is fast.
- Usage: DNS, media transfer.

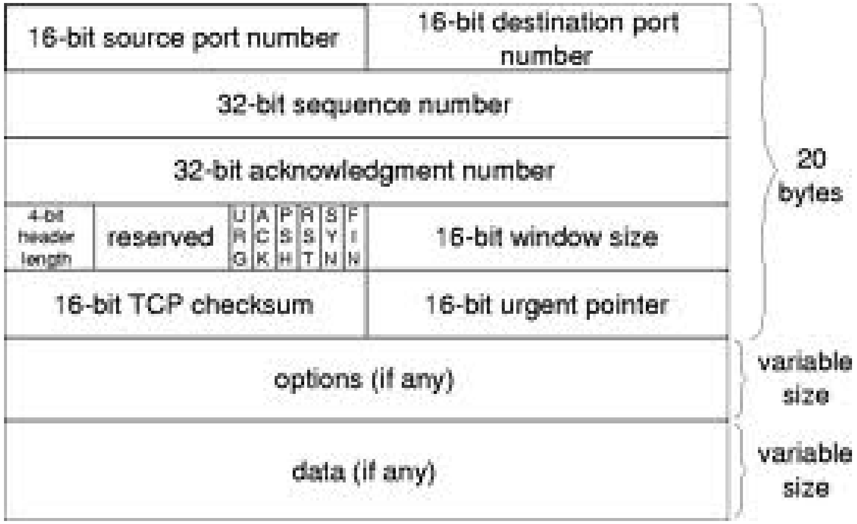


Source port number	Destination port number
Length	Checksum
Data	

TCP

- Transmission Control Protocol
- Connection-oriented
  - The involved processes first establish a connection, through handshaking, and then they exchange data.
- TCP offers full-duplex service.
  - Both processes can send data after the connection establishment.
- TCP offers point-to-point connection.
  - Only two remote processes take part in one connection.
- TCP offers reliable communication and congestion control.

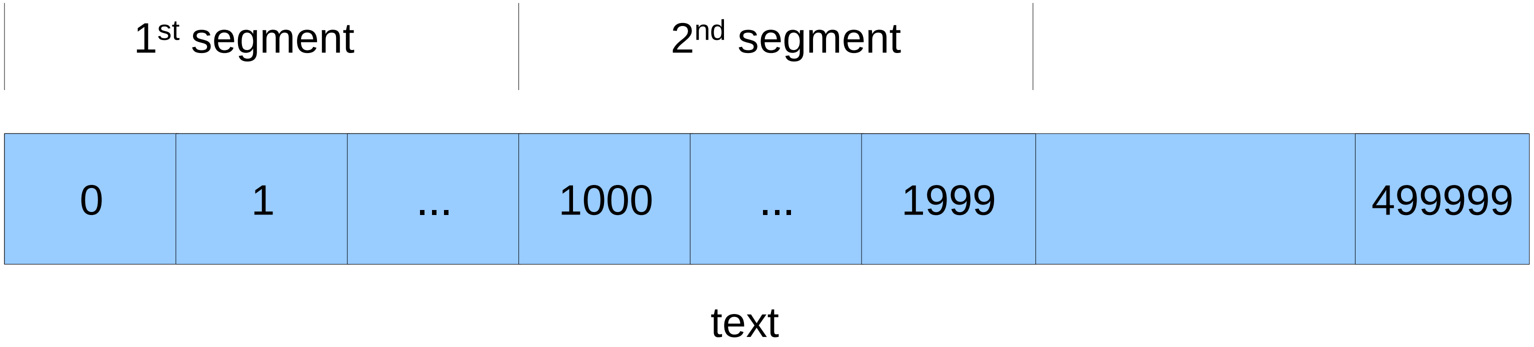
TCP segment



<table>
<tr><td colspan="8">16-bit source port number</td><td>16-bit destination port number</td><td rowspan="5">20 bytes</td></tr>
<tr><td colspan="9">32-bit sequence number</td></tr>
<tr><td colspan="9">32-bit acknowledgment number</td></tr>
<tr><td>4-bit header length</td><td>reserved</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>16-bit window size</td></tr>
<tr><td colspan="8">16-bit TCP checksum</td><td>16-bit urgent pointer</td></tr>
<tr><td colspan="9">options (if any)</td><td>variable size</td></tr>
<tr><td colspan="9">data (if any)</td><td>variable size</td></tr>
</table>

TCP

- Reliability → all the data reaches the destination
- The **destination** should **acknowledge** the received segments to the source.
- Every TCP segment has:
  - Sequence number = number of the first byte in the segment.
  - Acknowledgement number = number of the next byte that the host expects.



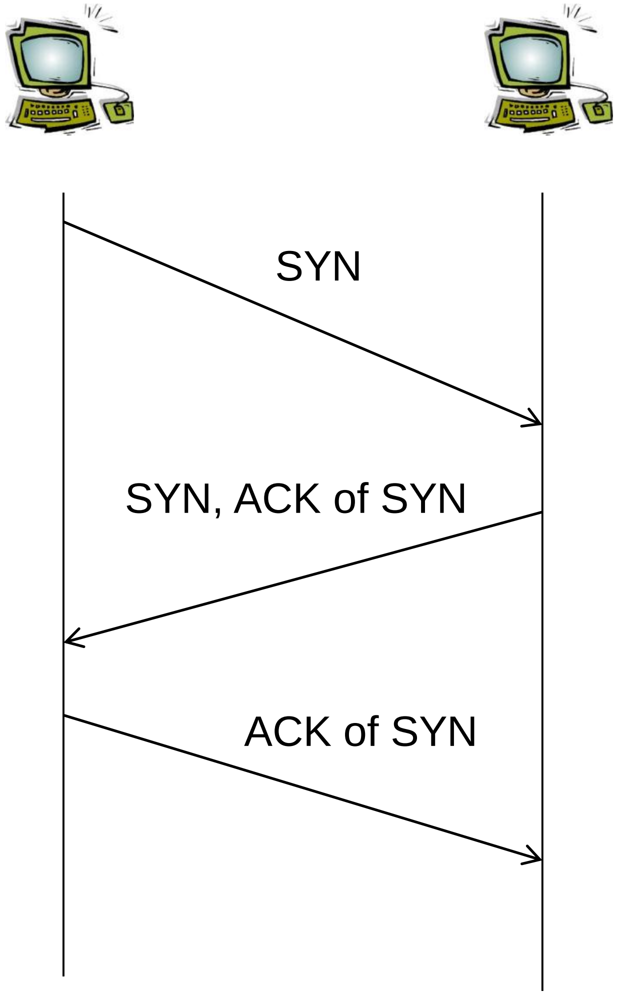
1st segment

2nd segment

0 | 1 | ... | 1000 | ... | 1999 | | 499999

text

TCP Handshake

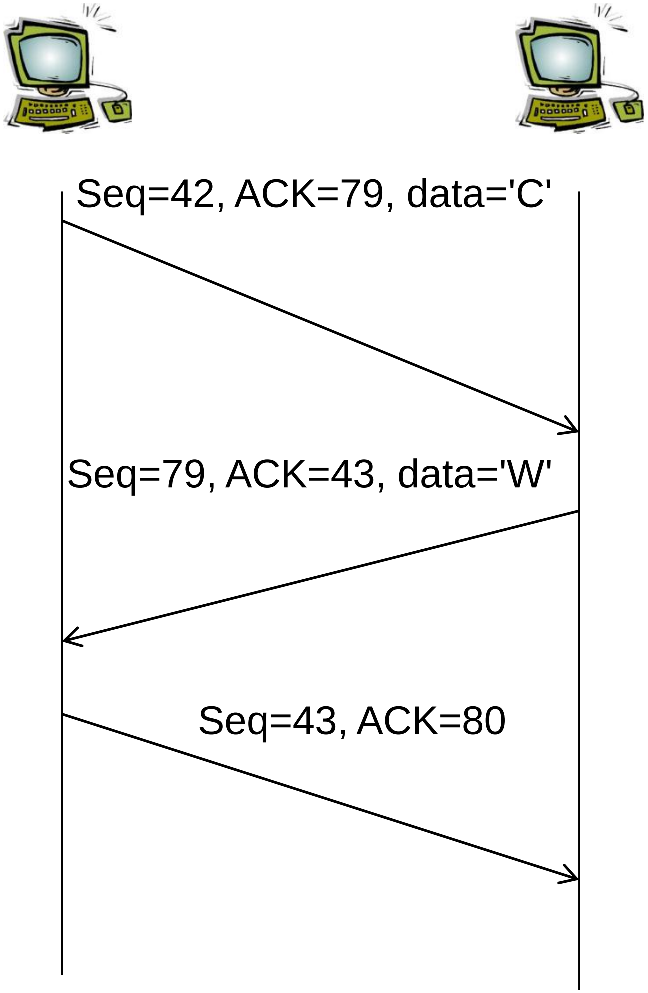


SYN

SYN, ACK of SYN

ACK of SYN

TCP

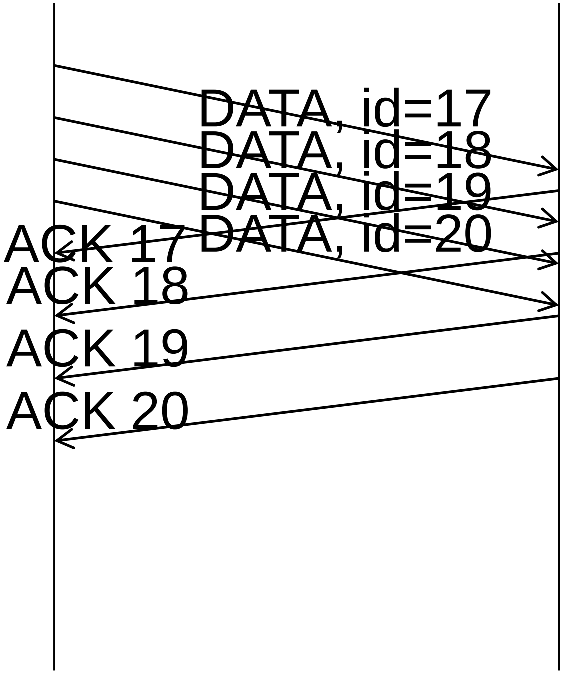


TCP: Retransmission

- What happens when a segment is lost or broken?
- No acknowledgment.
- The source waits for a specific time period and then it retransmits the segment.
- How long does it have to wait?

TCP Windows

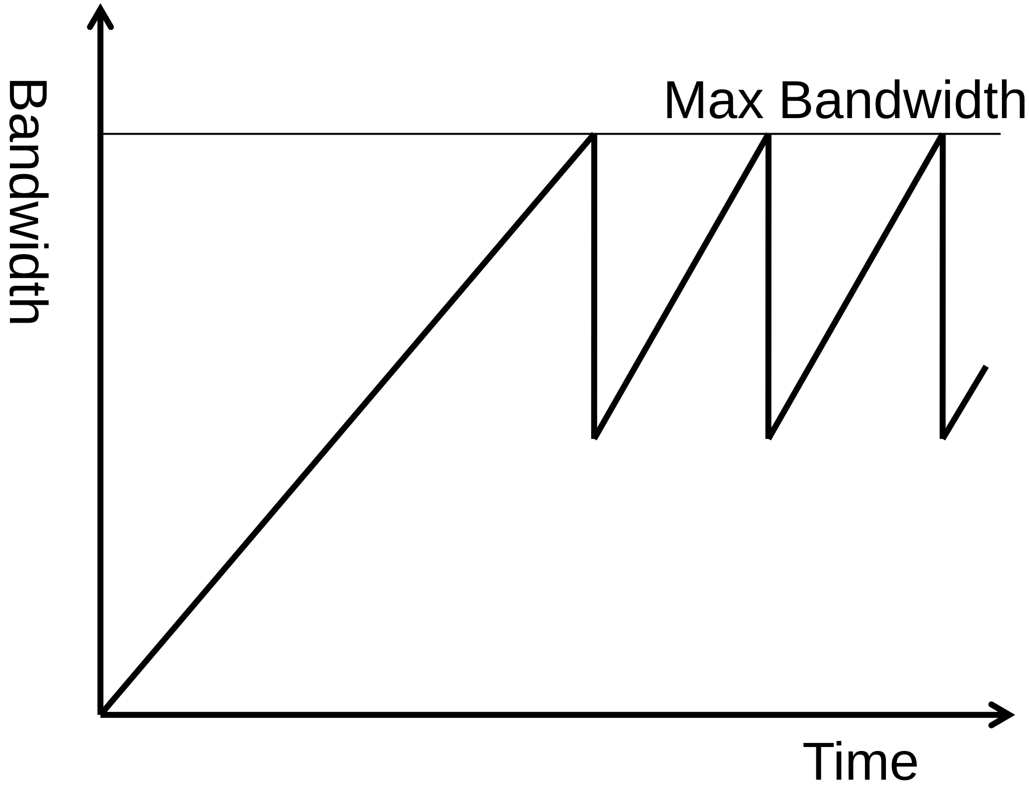
- Instead of waiting for the acknowledgment of one frame before sending the next one, the source should send a window of frames.



TCP Congestion Control

- TCP increases its window size as long as no packets are dropped (linearly).
- It halves the window size when a packet drop occurs.
  - A packet drop is evident from the absence of acknowledgements.
- Therefore, it will slowly build up to the max bandwidth, and hover around the max.
  - It doesn't achieve the max possible, though.
  - Instead, it shares the bandwidth well with other TCP connections

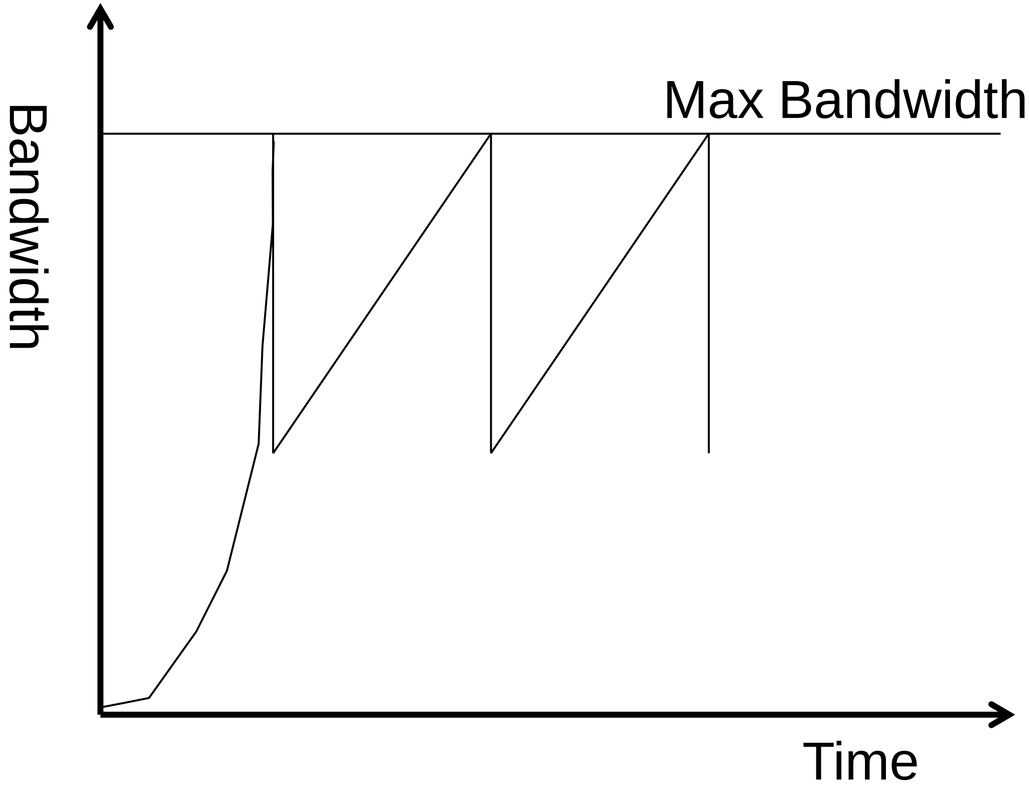
TCP Congestion Control



TCP Slow Start

- Linear increase takes a long time to build up a window size that matches the link bandwidth delay.
- Most file transactions are not long enough.
- Consequently, TCP can spend a lot of time with small windows, never getting the chance to reach a sufficiently large window size.
- Fix: Allow TCP to build up to a large window size **initially** by **doubling the window size until first loss**.

TCP Slow Start



Today

- Logical communication between remote processes.

Coming up…

- Next lecture: Application layer
- HW5: due on Wednesday
- Friday: review
- Next Monday: no class
- Next Tuesday: final exam
- Next Wednesday: no class
- Student evaluations: open today